Pinpoint分析报告

项目名称:	郓城农商-整存授信-src-android
首次分析时间:	2020年03月16日03:52:13
最近分析时间:	2020年03月19日05:43:19
报告范围:	单次分析
本次分析时间:	2020年03月19日05:43:19
标注范围:	未处理，可疑，确认

目录

项目总览

★★★★

质量等级

7000

实际代码行数

13

本次发现的总缺陷

13

最新发现缺陷数

0

较上次未检出的缺陷

代码度量

代码规模:

项目代码量:	8876 行
纯代码行:	7000 行

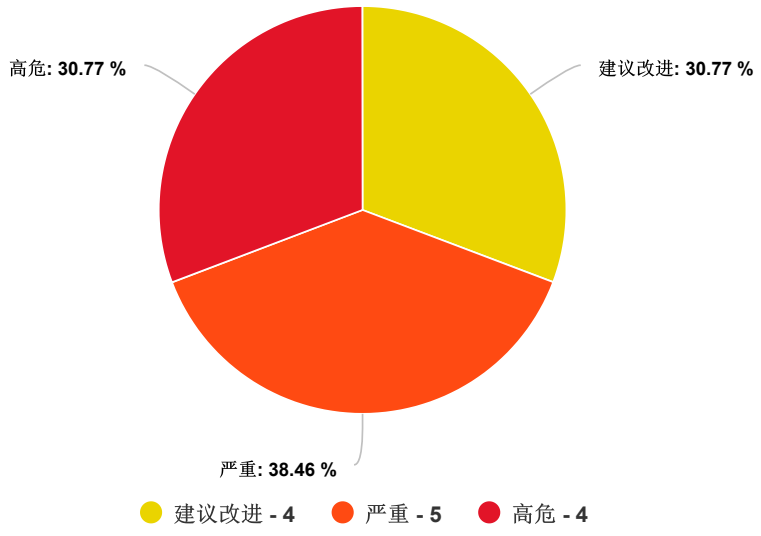
代码可维护性:

代码注释量	1010 行
注释覆盖率	11.379999999999999%
圈复杂度	293

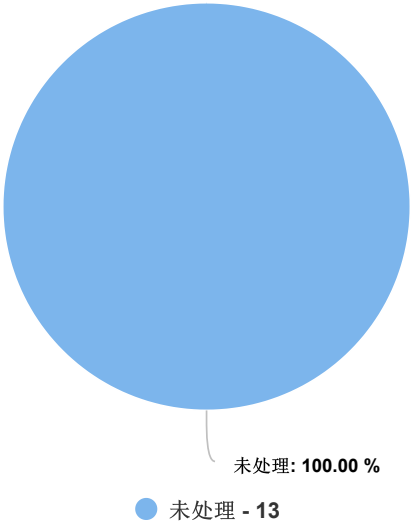
项目投入:

金钱:	51万
人力:	2.92 人
时间:	8.45 个月

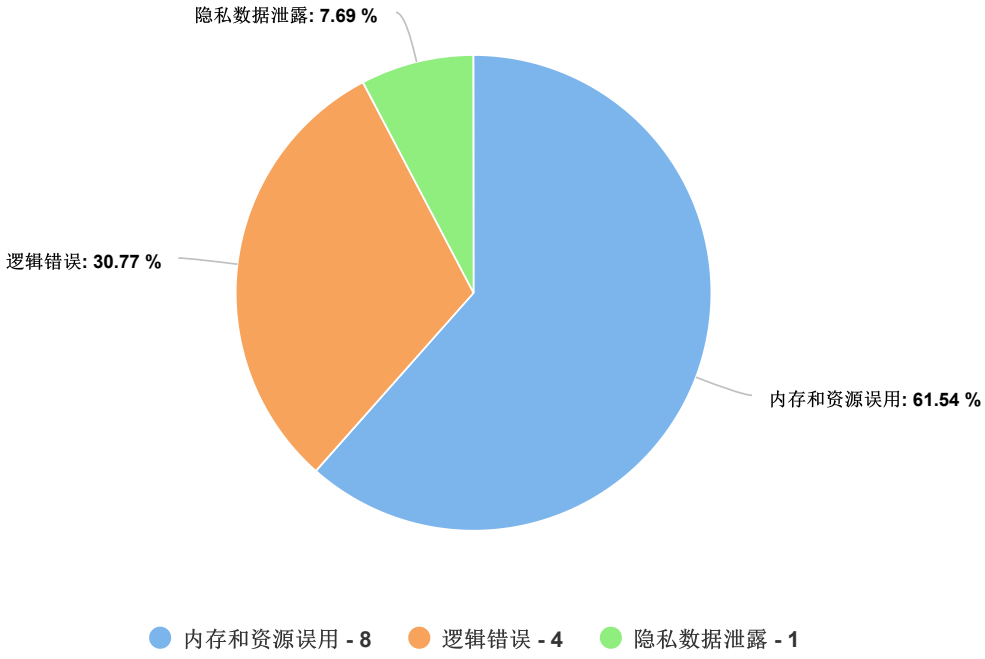
严重程度统计



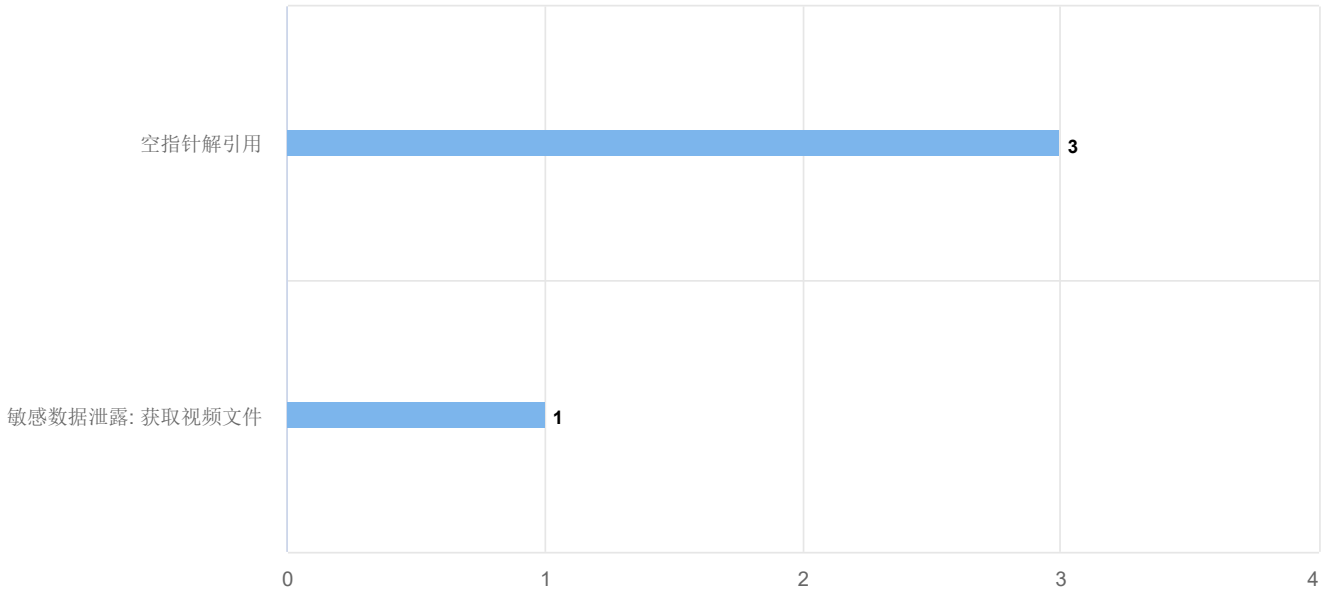
标注统计


未处理: 100.00 %
未处理 - 13

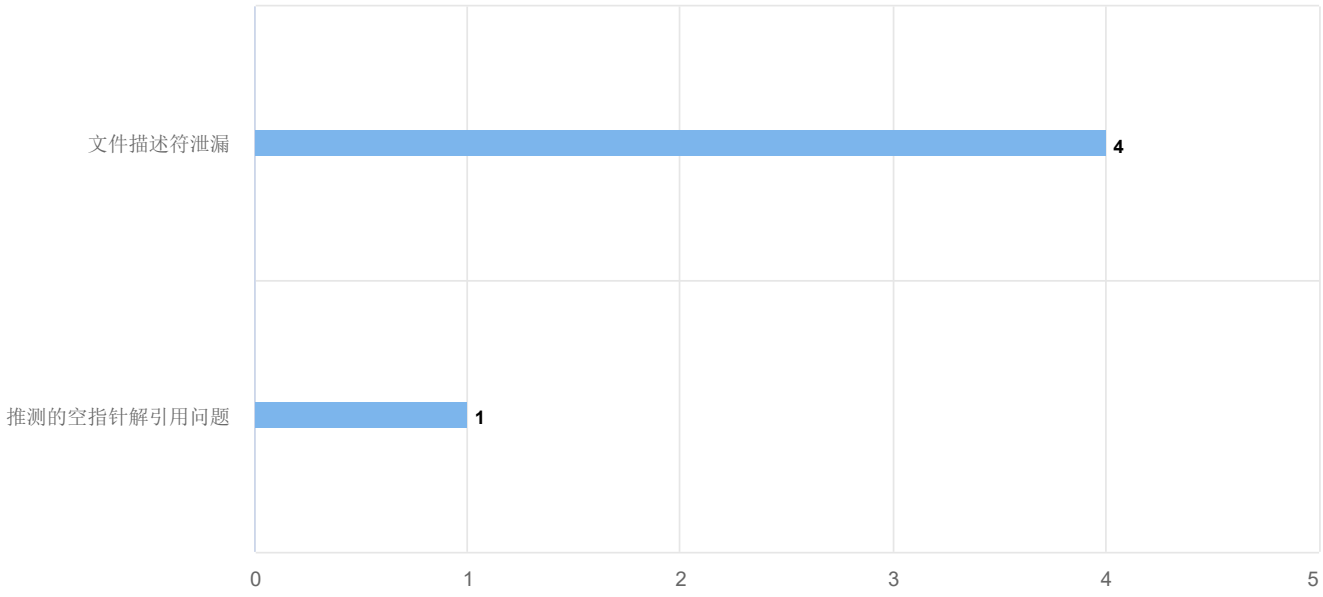
漏洞分类统计


隐私数据泄露: 7.69 %
逻辑错误: 30.77 %
内存和资源误用: 61.54 %
内存和资源误用 - 8
逻辑错误 - 4
隐私数据泄露 - 1

数量最多的十大高危缺陷类型



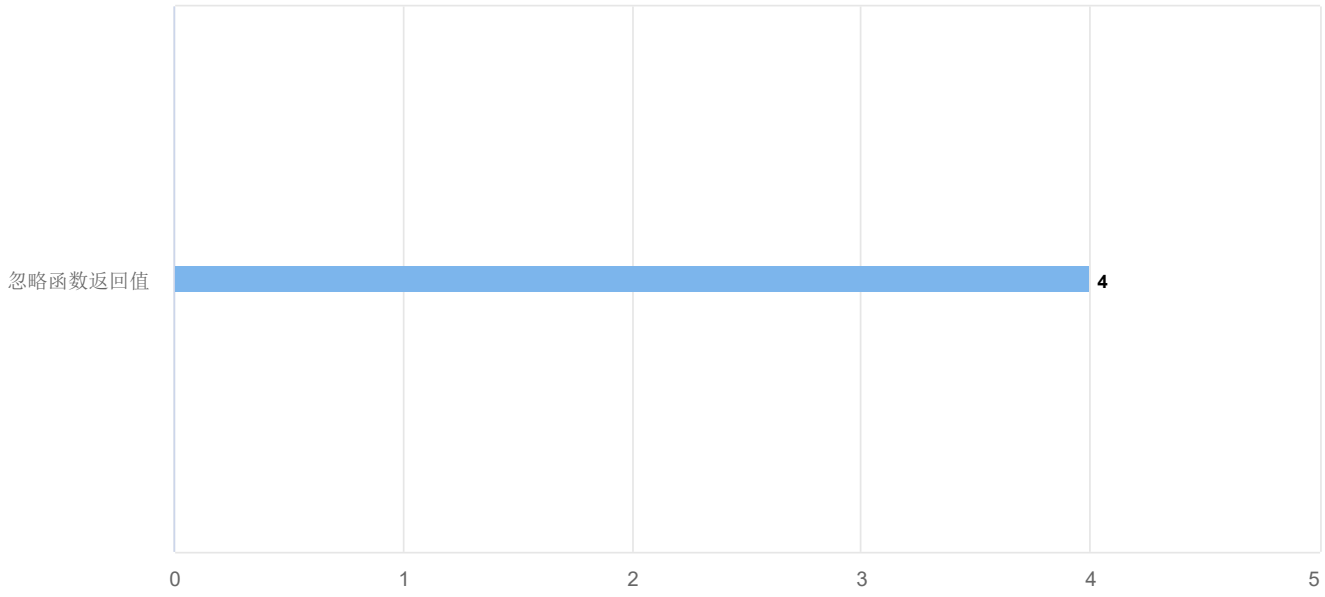
数量最多的十大严重缺陷类型



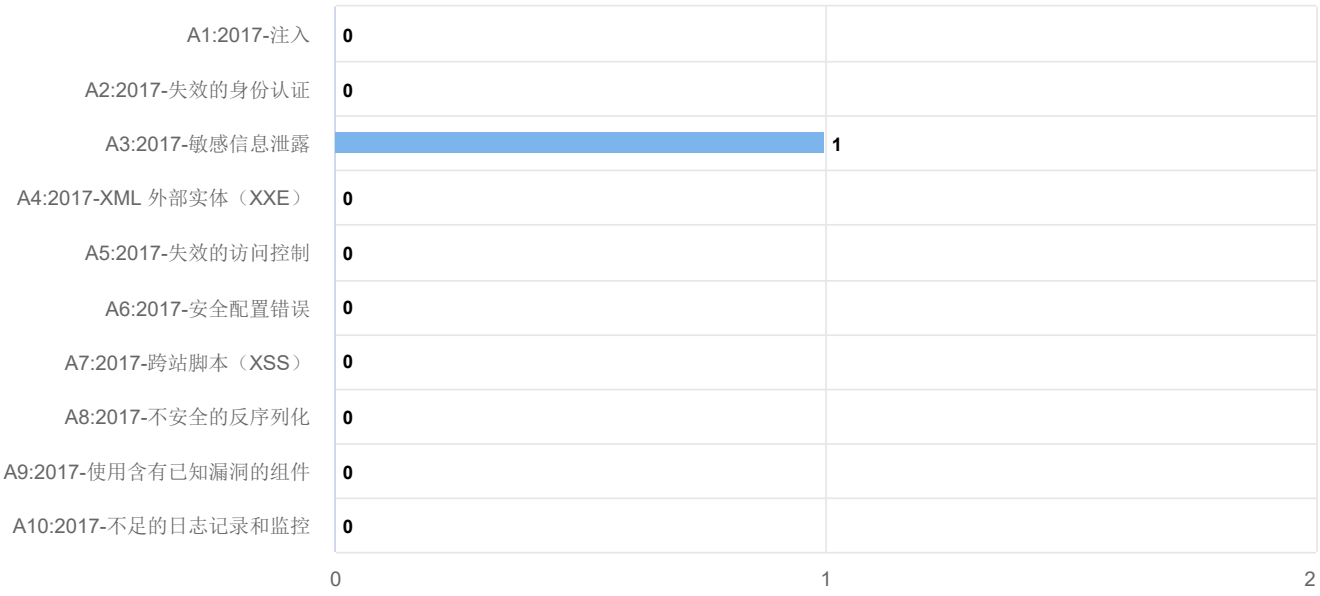
数量最多的十大一般缺陷类型

无数据

数量最多的十大建议改进缺陷类型



OWASP 十大严重安全漏洞



CWE/SANS 排名前25的严重安全漏洞

[高危] 敏感数据泄露: 获取视频文件

漏洞报告 (1 - 1)

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/utils.java: 157

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 9e5902fc1375316c08cb38a4200800c9

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
139             if (path != null) {
140                 path = Uri.decode(path);
141                 ContentResolver cr = context.getContentResolver();
142                 StringBuffer buff = new StringBuffer();
143                 buff.append("
("").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
"'").append(")");
```

1 函数**android.content.ContentResolver.query**执行完成并返回给调用者

```
144                 Cursor cur =
cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
142                 StringBuffer buff = new StringBuffer();
143                 buff.append("
("").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
"'").append(")");
144                 Cursor cur =
cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
```

2 可能的过滤处理：<param2>.getContentResolver().query(android.provider.MediaStore$Images$Media.EXTERNAL_CONTENT_URI,new(32),new java.lang.StringBuffer().toString(),null,null).isAfterLast()==0被用作分支条件，这可能是一个对不信任数据的危险性检查

```
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
143                 buff.append("
(").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
"'").append(")");
144                 Cursor cur =
cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
```

3 调用函数**android.content.ContentResolver.query**的返回值的**android.database.Cursor.getColumnIndex**方法

```
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
152                 }
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
152                 }
```

4 可能的过滤处理：$u3#10==0被用作分支条件，这可能是一个对不信任数据的危险性检查

```
153                 cur.close();
154                 if (index == 0) {
155                 } else {
156                     Uri u = Uri.parse("content://media/external/images/media/"
    + index);
157                     System.out.println("temp uri is :" + u);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
143                 buff.append("
    (").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
    "'").append(")");
144                 Cursor cur =
    cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
    MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
    buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
```

5 函数**android.database.Cursor.getColumnIndex**执行完成并返回给调用者

```
148                     index =
    cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
    cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
152                 }
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
144                 Cursor cur =
    cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
    MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
    buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
    cur.getColumnIndex(MediaStore.Images.ImageColumns. ID);
```

6 函数**android.database.Cursor.getColumnIndex**的返回值作为第1个参数传递给函数**android.database.Cursor.getInt**

```
149                     index = cur.getInt(index);
150                     dataIdx =
    cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
```

```
152                 }
153                 cur.close();
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
144                 Cursor cur =
cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns. ID);
```

7 函数**android.database.Cursor.getInt**执行完成并返回给调用者

```
149                     index = cur.getInt(index);
150                     dataIdx =
cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
152                 }
153                 cur.close();
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
142                 StringBuffer buff = new StringBuffer();
143                 buff.append("
(").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
"'").append(")");
144                 Cursor cur =
cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
```

8 可能的过滤处理：<param2>.getContentResolver().query(android.provider.MediaStore$Images$Media.EXTERNAL_CONTENT_URI,new(32),new java.lang.StringBuffer().toString(),null,null).isAfterLast()==0被用作分支条件，这可能是一个对不信任数据的危险性检查

```
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
142                 StringBuffer buff = new StringBuffer();
143                 buff.append("
("").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
"'").append(")");
144                 Cursor cur =
cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
```

9 在条件分支处走false分支 (**<param2>.getContentResolver().query(android.provider.MediaStore$Images$Media.EXTERNAL_CONTENT_URI,new(32),new java.lang.StringBuffer().toString(),null,null).isAfterLast()==0**为false)

```
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
151                     path = cur.getString(dataIdx);
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
```

10 函数**android.database.Cursor.getInt**的返回值作为第1个参数传递给函数 **java.lang.StringBuilder.append**

```
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
151                     path = cur.getString(dataIdx);
```

```
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
```

11 调用被分配的内存的**java.lang.StringBuilder.append**方法

```
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
151                     path = cur.getString(dataIdx);
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
```

12 调用被分配的内存的**java.lang.StringBuilder.toString**方法

```
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
151                     path = cur.getString(dataIdx);
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
```

13 函数**java.lang.StringBuilder.toString**的返回值作为第1个参数传递给函数**android.net.Uri.parse**

```
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
151                     path = cur.getString(dataIdx);
```

```
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
```

14 函数**android.net.Uri.parse**执行完成并返回给调用者

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
161                 return new File(path);
```

15 函数**android.net.Uri.parse**的返回值作为第1个参数传递给函数**java.lang.StringBuilder.append**

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
161                 return new File(path);
```

16 调用被分配的内存的**java.lang.StringBuilder.append**方法

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
152                 }
```

```
153                 cur.close();
154                 if (index == 0) {
155                 } else {
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
```

17 调用被分配的内存的**java.lang.StringBuilder.toString**方法

```
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
161                 return new File(path);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
152                 }
153                 cur.close();
154                 if (index == 0) {
155                 } else {
156                     Uri u = Uri.parse("content://media/external/images/media/"
+ index);
```

18 函数**java.lang.StringBuilder.toString**的返回值作为第1个参数传递给函数**java.io.PrintStream.println**

```
157                     System.out.println("temp uri is :" + u);
158                 }
159             }
160             if (path != null) {
161                 return new File(path);
```

[高危] 空指针解引用

漏洞报告

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/clutter.java: 231

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 445d6f5dddd58bd003e7aab14d52e16d

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
215     public void mergeFiles3(List<File> tempFiles, String amrFileName) {
216         try{
217             File realFile=new File("/sdcard/record/"+amrFileName+".amr");
218             FileOutputStream fos = new FileOutputStream(realFile);
219             RandomAccessFile ra = null;
        1 在条件分支处走false分支($u0#13<<param1>.size()为false)
220             for (int i = 0; i < tempFiles.size(); i++) {
221                 ra = new RandomAccessFile(tempFiles.get(i), "r");
222                 if (i != 0) {
223                     ra.seek(6);//跳过amr文件的文件头
224                 }
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
226                 int len = 0;
227                 while ((len = ra.read(buffer)) != -1) {
228                     fos.write(buffer, 0, len);
229                 }
230             }
        2 调用null的java.io.RandomAccessFile.close方法
231             ra.close();
232             fos.close();
233         } catch (Exception e) {
234             e.printStackTrace();
235         }
```

缺陷位置:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java: 260

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 838a7a6897d11c4ad93b6763dd90eb3d

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
244     public void uniteAMRFile(String[] partsPaths, String unitedFilePath) {
245         try {
246             File unitedFile = new
    File("/sdcard/record/"+unitedFilePath+".amr");
247             FileOutputStream fos = new FileOutputStream(unitedFile);
248             RandomAccessFile ra = null;
        1 在条件分支处走false分支($u0#12<<param1>.length为false)
249             for (int i = 0; i < partsPaths.length; i++) {
250                 ra = new RandomAccessFile(partsPaths[i], "r");
251                 if (i != 0) {
252                     ra.seek(6);
253                 }
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
255                 int len = 0;
256                 while ((len = ra.read(buffer)) != -1) {
257                     fos.write(buffer, 0, len);
258                 }
259             }
        2 调用null的java.io.RandomAccessFile.close方法
260             ra.close();
261             fos.close();
262         } catch (Exception e) {
263         }
264     }
```

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/getGps.java: 24

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: ab47cf112115586d64ee48061a344e74

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
124         if (context instanceof Activity) {
125             return (Activity) context;
126         }
127         if (context instanceof ContextWrapper) {
128             ContextWrapper wrapper = (ContextWrapper) context;
```

1 返回**null**给函数调用者

```
129             return findActivity(wrapper.getBaseContext());
130         } else {
131             return null;
132         }
133     }
```

YunChengAndroid.zip/app/src/main/java/Utils/getGps.java

```
19  //    监听
20      LocationListener locationListener;
21  //    GPS管理者
22      LocationManager locationManager;
23      public Location getGps(Context context){
```

2 调用函数**findActivity**的返回值的**android.app.Activity.getSystemService**方法（函数**findActivity**的返回值可能为空指针）

```
24          locationManager = (LocationManager)
utils.findActivity(context).getSystemService(LOCATION_SERVICE);
25          //权限检查,编辑器自动添加
26          if (ActivityCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
27              // TODO: Consider calling
28              //    ActivityCompat#requestPermissions
```

[严重] 推测的空指针解引用问题

漏洞报告 (1 - 1)

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/utils.java: 379

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: d57f6aafbd27b155f51f361cf1e348ab

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
370     }
371     //    判断当前值是否为null 返回boolean
372     public static String isNUllReturnBoolean(String str){
373         boolean is=false;
374         if (str==null){
    1 这里比较了str和null，说明str可能为空指针
375             is= false;
376
377         }
378         if (str.equals(null)){
379             is= false;
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
374         if (str==null){
375             is= false;
376
377         }
378         if (str.equals(null)){
    2 调用str的java.lang.String.equals方法
379             is= false;
380         }
381         if (str.equals("")){
382             is= false;
383         }
```

[严重] 文件描述符泄漏

漏洞报告

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/CrashHandler.java: 201

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 6cd8d4228a742238553580705adee79a

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/CrashHandler.java

```
194             String path = getGlobalpath();
195             File dir = new File(path);
196             if (!dir.exists())
197                 dir.mkdirs();
198
    1 创建了一个java.io.FileOutputStream类的对象，分配了一个文件描述符资源
199             FileOutputStream fos = new FileOutputStream(path + fileName, true);
200             fos.write(sb.getBytes());
201             fos.flush();
202             fos.close();
203         }
```

YunChengAndroid.zip/app/src/main/java/Utils/CrashHandler.java

```
196             if (!dir.exists())
197                 dir.mkdirs();
198
199             FileOutputStream fos = new FileOutputStream(path + fileName, true);
200             fos.write(sb.getBytes());
    2 在这里抛出异常，打开的文件描述符没有被正确关闭
201             fos.flush();
202             fos.close();
203         }
204         return fileName;
205     }
```

缺陷位置:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java: 228

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 99cfd92f199e886f6146e6c30528bbbc

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
216         try{
217             File realFile=new File("/sdcard/record/"+amrFileName+".amr");
218             FileOutputStream fos = new FileOutputStream(realFile);
219             RandomAccessFile ra = null;
220             for (int i = 0; i < tempFiles.size(); i++) {
```

1 创建了一个**java.io.RandomAccessFile**类的对象，分配了一个文件描述符资源

```
221                 ra = new RandomAccessFile(tempFiles.get(i), "r");
222                 if (i != 0) {
223                     ra.seek(6);//跳过amr文件的文件头
224                 }
225                 byte[] buffer = new byte[1024 * 8];
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
216         try{
217             File realFile=new File("/sdcard/record/"+amrFileName+".amr");
218             FileOutputStream fos = new FileOutputStream(realFile);
219             RandomAccessFile ra = null;
220             for (int i = 0; i < tempFiles.size(); i++) {
```

2 在条件分支处走true分支(**$u0#13!=0**为true)

```
221                 ra = new RandomAccessFile(tempFiles.get(i), "r");
222                 if (i != 0) {
223                     ra.seek(6);//跳过amr文件的文件头
224                 }
225                 byte[] buffer = new byte[1024 * 8];
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
222                 if (i != 0) {
223                     ra.seek(6);//跳过amr文件的文件头
224                 }
225                 byte[] buffer = new byte[1024 * 8];
226                 int len = 0;
```

3 在条件分支处走true分支(**new java.io.RandomAccessFile().read(new(32))!=(-1)**为true)

```
227                 while ((len = ra.read(buffer)) != -1) {
```

```
228                     fos.write(buffer, 0, len);
229                 }
230             }
231             ra.close();
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
223                     ra.seek(6);//跳过amr文件的文件头
224                 }
225                 byte[] buffer = new byte[1024 * 8];
226                 int len = 0;
227                 while ((len = ra.read(buffer)) != -1) {
```

4 在这里抛出异常，打开的文件描述符没有被正确关闭

```
228                     fos.write(buffer, 0, len);
229                 }
230             }
231             ra.close();
232             fos.close();
```

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/clutter.java: 257

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 8f3e99ca8ca8fd3ee4554be789d2c00a

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
245         try {
246             File unitedFile = new
    File("/sdcard/record/"+unitedFilePath+".amr");
247             FileOutputStream fos = new FileOutputStream(unitedFile);
248             RandomAccessFile ra = null;
249             for (int i = 0; i < partsPaths.length; i++) {
```

1 创建了一个**java.io.RandomAccessFile**类的对象，分配了一个文件描述符资源

```
250                 ra = new RandomAccessFile(partsPaths[i], "r");
251                 if (i != 0) {
252                     ra.seek(6);
253                 }
254                 byte[] buffer = new byte[1024 * 8];
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
245         try {
246             File unitedFile = new
    File("/sdcard/record/"+unitedFilePath+".amr");
247             FileOutputStream fos = new FileOutputStream(unitedFile);
248             RandomAccessFile ra = null;
249             for (int i = 0; i < partsPaths.length; i++) {
```

2 在条件分支处走true分支(**$u0#12!=0**为true)

```
250                 ra = new RandomAccessFile(partsPaths[i], "r");
251                 if (i != 0) {
252                     ra.seek(6);
253                 }
254                 byte[] buffer = new byte[1024 * 8];
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
251                 if (i != 0) {
252                     ra.seek(6);
253                 }
254                 byte[] buffer = new byte[1024 * 8];
255                 int len = 0;
```

3 在条件分支处走true分支(**new java.io.RandomAccessFile().read(new(32))!=(-1)**为true)

```
256                 while ((len = ra.read(buffer)) != -1) {
257                     fos.write(buffer, 0, len);
258                 }
259             }
260             ra.close();
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
252                     ra.seek(6);
253                 }
254                 byte[] buffer = new byte[1024 * 8];
255                 int len = 0;
256                 while ((len = ra.read(buffer)) != -1) {
```

4 在这里抛出异常，打开的文件描述符没有被正确关闭

```
257                     fos.write(buffer, 0, len);
258                 }
259             }
260             ra.close();
261             fos.close();
```

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/clutter.java: 302

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 0fc27afcff1bd51a25ea9edb403f1c3b

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
271         if (tempFiles.isEmpty()) return;//如果还没录制则不进行合并
272         File realFile=getFile(false);
273         try {
274             FileOutputStream fos=new FileOutputStream(realFile);
275             for (int i = 0; i < tempFiles.size(); i++) {//遍历tempFiles集合，合并所有临时文件
```

1 创建了一个**java.io.FileInputStream**类的对象，分配了一个文件描述符资源

```
276                 FileInputStream fis=new FileInputStream(tempFiles.get(i));
277                 byte[] tmpBytes = new byte[fis.available()];
278                 int length = tmpBytes.length;//文件长度
279                 //头文件
280                 if(i==0){
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
273         try {
274             FileOutputStream fos=new FileOutputStream(realFile);
275             for (int i = 0; i < tempFiles.size(); i++) {//遍历tempFiles集合，合并所有临时文件
276                 FileInputStream fis=new FileInputStream(tempFiles.get(i));
277                 byte[] tmpBytes = new byte[fis.available()];
```

2 在条件分支处走true分支(**$u3#8==0**为true)

```
278                 int length = tmpBytes.length;//文件长度
279                 //头文件
280                 if(i==0){
281                     while(fis.read(tmpBytes)!=-1){
282                         fos.write(tmpBytes,0,length);
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
276                 FileInputStream fis=new FileInputStream(tempFiles.get(i));
277                 byte[] tmpBytes = new byte[fis.available()];
278                 int length = tmpBytes.length;//文件长度
279                 //头文件
280                 if(i==0){
```

3 在条件分支处走true分支(**new java.io.FileInputStream().read(new(32))!=(-1)**为true)

```
281                     while(fis.read(tmpBytes)!=-1){
```

```
282                         fos.write(tmpBytes,0,length);
283                     }
284                 }
285                 //之后的文件，去掉头文件就可以了.amr格式的文件的头信息为 6字节
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
277                 byte[] tmpBytes = new byte[fis.available()];
278                 int length = tmpBytes.length;//文件长度
279                 //头文件
280                 if(i==0){
281                     while(fis.read(tmpBytes)!=-1){
```

4 在这里抛出异常

```
282                         fos.write(tmpBytes,0,length);
283                     }
284                 }
285                 //之后的文件，去掉头文件就可以了.amr格式的文件的头信息为 6字节
286                 else{
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
297         } catch (Exception e) {
298             // TODO Auto-generated catch block
299             e.printStackTrace();
300         }
301         //删除合并过的临时文件
```

5 在条件分支处走true分支(**<param1>.iterator().hasNext()!=0**为true)

```
302         for (File file:tempFiles) {
303             if (file.exists()) {
304                 file.delete();
305             }
306         }
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
298             // TODO Auto-generated catch block
299             e.printStackTrace();
300         }
301         //删除合并过的临时文件
302         for (File file:tempFiles) {
```

6 在条件分支处走true分支(**<param1>.iterator().next().exists()!=0**为true)

```
303             if (file.exists()) {
304                 file.delete();
305             }
306         }
307     }
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
297         } catch (Exception e) {
298             // TODO Auto-generated catch block
299             e.printStackTrace();
300         }
301         //删除合并过的临时文件
```

7 在条件分支处走false分支(**<param1>.iterator().hasNext()!=0**为false)，打开的文件描述符没有被正确关闭

```
302         for (File file:tempFiles) {
303             if (file.exists()) {
304                 file.delete();
305             }
306         }
```

[建议改进] 忽略函数返回值

漏洞报告

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/clutter.java: 277

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 904c012913705665b33f1c8705f052b6

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
272         File realFile=getFile(false);
273         try {
274             FileOutputStream fos=new FileOutputStream(realFile);
275             for (int i = 0; i < tempFiles.size(); i++) {//遍历tempFiles集合，合并所有临时文件
276                 FileInputStream fis=new FileInputStream(tempFiles.get(i));
```

1 函数 **java.io.FileInputStream.available** 的返回值在1次中被检查1次，比如这里

```
277                 byte[] tmpBytes = new byte[fis.available()];
278                 int length = tmpBytes.length;//文件长度
279                 //头文件
280                 if(i==0){
281                     while(fis.read(tmpBytes)!=-1){
```

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
272         File realFile=getFile(false);
273         try {
274             FileOutputStream fos=new FileOutputStream(realFile);
275             for (int i = 0; i < tempFiles.size(); i++) {//遍历tempFiles集合，合并所有临时文件
276                 FileInputStream fis=new FileInputStream(tempFiles.get(i));
```

2 函数 **java.io.FileInputStream.available** 的返回值在传给其他函数前没有被有作用的检查

```
277                 byte[] tmpBytes = new byte[fis.available()];
278                 int length = tmpBytes.length;//文件长度
279                 //头文件
280                 if(i==0){
281                     while(fis.read(tmpBytes)!=-1){
```

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/utils.java: 147

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: 1fe138edd7cfe6eda141701c764edd87

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
363     public static Uri getImageContentUri(Context context,File imageFile) {
364         String filePath = imageFile.getAbsolutePath();
365         Cursor cursor =
context.getContentResolver().query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI
,
366                 new String[] { MediaStore.Images.Media._ID },
MediaStore.Images.Media.DATA + "=? ",
367                 new String[] { filePath }, null);
```

1 函数 **android.database.Cursor.moveToFirst** 的返回值在3次中被检查2次，比如这里

```
368         if (cursor != null && cursor.moveToFirst()) {
369             int id =
cursor.getInt(cursor.getColumnIndex(MediaStore.MediaColumns._ID));
370             Uri baseUri = Uri.parse("content://media/external/images/media");
371             return Uri.withAppendedPath(baseUri, "" + id);
372         } else {
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
162             }
163         } else if ("content".equals(uri.getScheme())) {
164             // 4.2.2以后
165             String[] proj = { MediaStore.Images.Media.DATA };
166             Cursor cursor = context.getContentResolver().query(uri, proj, null,
null, null);
```

2 另一个检查的例子

```
167             if (cursor.moveToFirst()) {
168                 int columnIndex =
cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
169                 path = cursor.getString(columnIndex);
170             }
171             cursor.close();
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
142                 StringBuffer buff = new StringBuffer();
143                 buff.append("
    (").append(MediaStore.Images.ImageColumns.DATA).append("=").append("'" + path +
    "'").append(")");
144                 Cursor cur =
    cr.query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[] {
    MediaStore.Images.ImageColumns._ID, MediaStore.Images.ImageColumns.DATA },
    buff.toString(), null, null);
145                 int index = 0;
146                 int dataIdx = 0;
```

3 函数 **android.database.Cursor.moveToFirst** 的返回值没有被检查

```
147                 for (cur.moveToFirst(); !cur.isAfterLast(); cur.moveToNext()) {
148                     index =
    cur.getColumnIndex(MediaStore.Images.ImageColumns._ID);
149                     index = cur.getInt(index);
150                     dataIdx =
    cur.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
151                     path = cur.getString(dataIdx);
```

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/utils.java: 250

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: dc39adaf58cc6bdf83e2a1d112a936e6

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
241     public static String removeChar(String str, String remove){
242         String newStr = "";
243         boolean match = false;
244         for(int i = 0; i<str.length(); i++){
245             for(int j=0; j<remove.length(); j++){
```

1 函数 **java.lang.String.charAt** 的返回值在12次中被检查8次，比如这里

```
246                 if(str.charAt(i) == remove.charAt(j))
247                     match = true;
248             }
249             if(match == false)
250                 newStr = newStr + str.charAt(i);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
241     public static String removeChar(String str, String remove){
242         String newStr = "";
```

```
243         boolean match = false;
244         for(int i = 0; i<str.length(); i++){
245             for(int j=0; j<remove.length(); j++){
```

2 另一个检查的例子

```
246                 if(str.charAt(i) == remove.charAt(j))
247                     match = true;
248             }
249             if(match == false)
250                 newStr = newStr + str.charAt(i);
```

YunChengAndroid.zip/app/src/main/java/com/yuncheng/administrator/yunchengandroid/HangYeXiFen.java

```
213     public static String removeChar(String str, String remove){
214         String newStr = "";
215         boolean match = false;
216         for(int i = 0; i<str.length(); i++){
217             for(int j=0; j<remove.length(); j++){
```

3 另一个检查的例子

```
218                 if(str.charAt(i) == remove.charAt(j))
219                     match = true;
220             }
221             if(match == false)
222                 newStr = newStr + str.charAt(i);
```

YunChengAndroid.zip/app/src/main/java/com/yuncheng/administrator/yunchengandroid/LouPanBiao.java

```
115     public static String removeChar(String str, String remove){
116         String newStr = "";
117         boolean match = false;
118         for(int i = 0; i<str.length(); i++){
119             for(int j=0; j<remove.length(); j++){
```

4 另一个检查的例子

```
120                 if(str.charAt(i) == remove.charAt(j))
121                     match = true;
122             }
123             if(match == false)
124                 newStr = newStr + str.charAt(i);
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
245             for(int j=0; j<remove.length(); j++){
246                 if(str.charAt(i) == remove.charAt(j))
247                     match = true;
248             }
```

```
249             if(match == false)
```

5 函数 **java.lang.String.charAt** 的返回值在传给其他函数前没有被有作用的检查

```
250                 newStr = newStr + str.charAt(i);
251             match = false;
252         }
253         return newStr;
254     }
```

缺陷位置:
YunChengAndroid.zip/app/src/main/java/Utils/utils.java: 382

标注: 未处理

时间: 2020-03-16 17:43:19

缺陷ID: adc4867eade584205fd44e63e5031e22

触发步骤:

YunChengAndroid.zip/app/src/main/java/Utils/clutter.java

```
312      * */
313     private File getFile(boolean isTempFile) {
314         // TODO Auto-generated method stub
315         finalFile=null;
316         if (!Environment.getExternalStorageState().
```

1 函数 **java.lang.String.equals** 的返回值在76次中被检查73次，比如这里

```
317                 equals(Environment.MEDIA_MOUNTED)) {
318             Log.w("Waring", "检测到你的手机没有插入SD卡，请插入SD后再试！");
319         }
320         //获取系统的24小时制时间作为文件名(HH为24小时制，hh为12小时制)
321         SimpleDateFormat simpleDateFormat=new SimpleDateFormat(
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
387         return  is?str:"0";
388     }
389 //     正在用的防止給下拉框赋时为Null
390     public static int NoNull(String str){
391         String Index= !(str.equals("null"))?str:String.valueOf(0);
```

2 另一个检查的例子

```
392         if (Index.equals("0")){
393             return 0;
394         }else if(Integer.parseInt(Index)>0){
395             return Integer.parseInt(Index)-1;
396         }
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
388     }
389 //     正在用的防止給下拉框赋时为Null
390     public static int NoNull(String str){
391         String Index= !(str.equals("null"))?str:String.valueOf(0);
392         if (Index.equals("0")){
    3 另一个检查的例子
393             return 0;
394         }else if(Integer.parseInt(Index)>0){
395             return Integer.parseInt(Index)-1;
396         }
397         return 0;
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
397         return 0;
398     }
399     //     正在用的防止給下拉框赋时为Null
400     public static int NoNullJIa(String str){
401         String Index= !(str.equals("null"))?str:String.valueOf(0);
    4 另一个检查的例子
402         return Integer.parseInt(Index);
403     }
404 //
405     public static int woHangDaiKuan(String str){
406         String Index =str.equals("null")?"0":str;
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
354         if (str==null){
355             return "";
356 
357         }
358         if (str.equals(null)){
    5 另一个检查的例子
359             return "";
360 
361         }
362         if (str.equals("")){
363             return "";
```

YunChengAndroid.zip/app/src/main/java/Utils/utils.java

```
377         }
378         if (str.equals(null)){
```

```
379             is= false;
380         }
381         if (str.equals("")){
```

6 函数 **java.lang.String.equals** 的返回值没有被检查

```
382             is= false;
383         }
384         if (str.equals("null")){
385             is= false;
386         }
```